Java의 주요 특성

변석우

2019년

1 타입

- static types : 모든 변수는 타입을 가지며, 그 타입은 수행전(컴파일시에) 점검한다.
- primitive types (8개) : `byte, short, int, long, float, double, boolean, char`
- reference types : primitive type이 아닌 타입 (class 타입). 이 타입의 변수는 C의 포인터 처럼 메모리의 주소를 갖는다.
- class vs instace 변수; `static` 키워드로 선언된 변수는 class 변수, 그렇지 않는 변수들은 instance 변수이다. 클래스 변수는 `Class.var` 형태로, 인스턴스 변수는 `obj.var` 형태로 접근된다.
- 한 클래스로부터 여러 개의 객체들이 생성될 수 있는데, 클래스 변수는 모든 객체들에 의해서 공유되며 (메모리 공유), 인스턴스 변수는 각 객체마다 독립적인 메모리를 갖게 된다 (슬라이드 참조).
- 한 객체가 클래스로부터 `new` 의 의해서 생성될 때 그 객체는 클래스 타입 (reference 타입)을 갖게 되므로, 사실상 그 객체는 메모리를 포인트하고 있다. 따라서 한 객체를 다른 객체에 할당할 경우, 두 객체는 동일한 메모리 공간을 포인트 (공)하게 된다.
- 연습문제: 다음 프로그램의 수행 결과 출력되는 내용은 무엇인가?

```
class M {
    static int s;
    int a;
    public M() {
        s = 2;
        a = 5;
    }

    public int r() {
        s = s + 2;
        a = this.a + 1;
        return s + this.a;
    }
}

public class T {
    public static void main (String argv[]) {
        M m = new M();
        M n = new M();
        M.s = 4;
        System.out.println(m.r()+" "+m.s+" "+M.s+" "+n.r()+" "+n.s+" "+M.s);
        m = n;
        System.out.println(m.r()+" "+m.s+" "+M.s+" "+n.r()+" "+n.s+" "+M.s);
    }
}
```

- `this` : 모든 인스턴스 변수 및 인스턴스 메소드는 특정 객체에 소속되어 있으므로, 이들을 접근할 때는 반드시 이들이 소속된 객체와 함께 구분되어야 한다 (`obj.instVar` 의 형태), 인스턴스 변수에 객체가 명시되지 않는 경우 그 객체는 `this` 생성된 객체 자체를 의미한다. 즉, `this` 는 특정 객체가 아닌 동적으로 생성된 객체 자신을 의미한다.

2 Instance vs Class Methods

(슬라이드 참조)

3 Interface (너마지는 슬라디으 참조)

```
import java.util.ArrayList;
import java.util.Iterator;

interface I {
    void ma();
}

class A implements I {
    public void ma() {
        System.out.println("Implementing in A");
    }
}

class B implements I {
    public void ma() {
        System.out.println("Implementing in B");
    }
}

class C implements I {
    public void ma() {
        System.out.println("Implementing in C");
    }
}

class D implements I {
    public void ma() {
        System.out.println("Implementing in D");
    }
}

public class InterfaceTest {
    public static void main(String[] args) {
        // array whose elements are objects of subclasses implementing I
        ArrayList<I> arrs = new ArrayList<> ();
        arrs.add(new D());
        arrs.add(new B());
        arrs.add(new A());
        arrs.add(new C());
        for (I obj : arrs)
            obj.ma();
```

```
    }
}
```

4 Generics

- 다음 Select 를래스에서 k 메소드는 주어지는 두 개의 인수 중에서 첫 번째 인수를 선택하는 기능을 한다. 이 메소드는 타입 변수를 이용하여 정의되었으므로 여러 타입의 데이터를 인수로 받아 처리함을 볼 수 있다. 만약 타입 변수를 사용하지 않는다면 다양한 타입의 인수들의 처리를 어떻게 처리할 수 있는가? 타입 변수를 이용함으로써 어떤 이득을 얻는가?

```
class Select {
    static <T1,T2> T1 k(T1 x, T2 y) {
        return x;
    }

    public static void main(String[] args) {
        System.out.println("k(true, 1) is " + k(true, 1));
        System.out.println("k('a', false) is " + k('a', false));
        System.out.println("k(1, 'c') is " + k(1, 'c'));
    }
}
```

– 타입 변수를 이용하지 않는 경우, 각각의 입력에 따라 그에 대한 메소드를 정의해야 하며, 여러 개의 메소드가 정의됨에 따라 코드의 양이 늘어나는 문제가 생기게 된다.